Guide-Copy: Fast and Silent Migration of Virtual Machine for Datacenters

Jihun Kim, Dongju Chae, Jangwoo Kim, and Jong Kim
Department of Computer Science and Engineering, POSTECH
{jihun735, torr55, jangwoo, jkim}@postech.ac.kr

ABSTRACT

Cloud infrastructure providers deploy Dynamic Resource Management (DRM) to minimize the cost of datacenter operation, while maintaining the Service Level Agreement (SLA). Such DRM schemes depend on the capability to migrate virtual machine (VM) images. However, existing migration techniques are not suitable for highly utilized clouds due to their latency and bandwidth critical memory transfer mechanisms. In this paper, we propose guide-copy migration, a novel VM migration scheme to provide a fast and silent migration, which works nicely under highly utilized clouds. The guide-copy migration transfers only the memory pages accessed at the destination node in the near future by running a guide version of the VM at the source node and a migrated VM at the destination node simultaneously during the migration. The guide-copy migration's highly accurate and low-bandwidth memory transfer mechanism enables a fast and silent VM migration to maintain the SLA of all VMs in the cloud.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Distributed Systems*

General Terms

Experimentation, Performance

Keywords

Virtual Machines, Operating Systems, Process Migration

1. INTRODUCTION

Cloud infrastructure providers use a virtual machine (VM) migration to dynamically relocate VMs in the cloud [4, 11, 18, 26], in order to improve the performance of VMs and the overall utilization of a datacenter. For example, cloud providers can migrate a resource-critical VM to a higher performance node to improve the performance, move many VMs into a single physical node to reduce the cost of datacenter operation, and evict performance-contending VMs out of the busy node to avoid disturbing the performance of other VMs running in the node. Therefore, it is critical for cloud providers to be able to perform a fast and silent VM migration to maintain the Service Level Agreement (SLA) between all users and the cloud provider while minimizing the cost of datacenter operation.

A cloud-ready fast and silent VM migration must satisfy two fundamental requirements to maintain the SLAs of all users in the cloud. First, a VM migration should be performed fast enough to avoid exposing the migration latency to the user. Such a user-unnoticeable fast migration is often called a transparent live migration. Second, a VM migration should not disturb the performance of other VMs running in the cloud. We define such an undisturbing migration as a *silent* migration.

However, existing migration schemes cannot provide a fast and silent migration, which makes them inefficient for highly utilized cloud environments. First, existing migration schemes can become very slow or even fail, depending on the behavior of workloads. Furthermore, they completely ignore the migration's impacts on the performance of other VMs in the cloud. In fact, existing migration schemes are designed to take critical resources such as network bandwidth away from other VMs in the cloud.

Existing migration schemes can be broadly classified as either pre-copy schemes [7, 11] or post-copy schemes [14], based on when the VM context and its memory image are transferred. The pre-copy scheme transfers the VM's memory state before transferring the VM context, whereas the post-copy scheme transfers the VM context first and then transfers its memory state. To avoid exposing the long memory transfer latency, both schemes overlap the execution of the VM context and its memory transfer process as much as possible while consuming the entire network bandwidth. We define such a full-bandwidth consuming migration as a *loud* migration.

Both schemes must stop their execution to transfer memory pages updated during the pre-copy period or to resolve remote page faults caused during the post-copy period. As a result, the pre-copy scheme can cause a very slow migration or even a migration failure for write-intensive workloads, while the post-copy scheme can cause a very slow migration for memory-intensive workloads. Furthermore, both schemes can incur significant performance overhead to other VMs running in the cloud by consuming the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SC 13 November 17-21, 2013, Denver, CO, USA
Copyright 2013 ACM 978-1-4503-2378-9/13/11 ...$15.00.

entire network bandwidth available [25]. Therefore, existing schemes do not provide a fast and silent migration, which makes them inefficient for highly utilized clouds where the network bandwidth is one of the most performance-critical resources.

In this paper, we propose *guide-copy* migration, a novel VM migration scheme to provide a fast and silent migration, which works nicely under highly utilized clouds. The guide-copy scheme achieves a fast migration by maximizing the overlapping of VM execution and its memory copy process, and a silent migration by minimizing the network bandwidth usage. The main idea of guide-copy scheme is to transfer the VM context immediately to the destination node on a migration request, keep running the remaining VM context at the source node to identify memory pages accessed in the near future, and transfer them to the context at the destination node before being requested. We define the VM context at the source node as the '*guide context*' and the VM context at the destination node as the '*migrated context*'.

By arranging the guide context of the VM at the source node to run slightly ahead of the migrated context at the destination node, the guide-copy migration can identify performance-critical memory pages accurately and transfer them timely. The guide context of the VM does not affect the functionality of the migrated context at the destination node. In this way, the guide-copy migration hides the long memory copy latency, while minimizing the network bandwidth usage. The guide-copy migration's highly accurate and low-bandwidth memory transfer mechanism enables a fast and silent VM migration, which can be effectively used to maintain the SLAs of all VMs in the cloud.

We implement our guide-copy scheme on top of KVM [17] hypervisor running on Intel Xeon X5650 nodes connected by 1Gbps or 5Gbps network. Our example implementation shows that the guide-copy scheme can reduce the performance overhead of a conventional full-bandwidth migration by ~75% and the network-bandwidth overhead by ~80%, on average for a subset of SPEC CPU2006 benchmarks [13]. We also provide similar improvements for PARSEC multi-threaded benchmarks [6]. To the best of our knowledge, the guide-copy scheme is the first VM migration scheme to target the undisturbed performance of all VMs in the cloud by providing a fast and silent migration capability, which works nicely for the highly utilized clouds.

Our work makes the following contributions:

- **Novel design.** We propose the *guide-copy* migration scheme which uses a novel simultaneous VM execution model with an adaptive memory copy mechanism to maintain the performance of all VMs in the cloud for the first time.
- **Fast migration.** We show that the guide-copy scheme reduces the migration latency significantly by eliminating most of the performance-critical remote page faults.
- **Silent migration.** We show that the guide-copy scheme adaptively controls the memory transfer rate to avoid disturbing the performance of VMs in the cloud.
- **Strong results.** Our guide-copy implementation on top of KVM shows a significant performance improvement and network usage reduction compared to the existing migration schemes.

In Section 2, we discuss conventional VM migration schemes and their limitations. In Section 3, we introduce our guide-copy migration scheme, and Section 4 discusses our guide-copy implementation. Section 5 evaluates the high performance and low-bandwidth usage of the guide-copy scheme. Section 6 summarizes the related work and Section 7 draws conclusions.

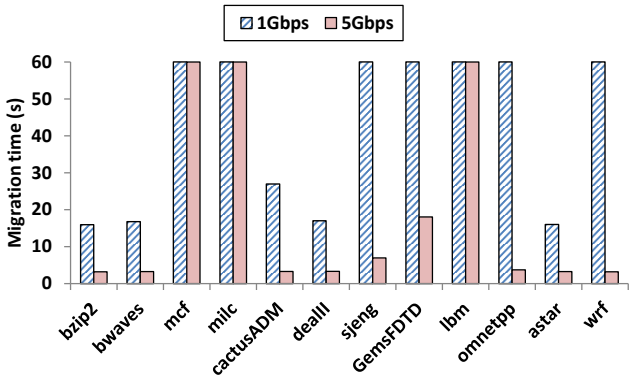


Figure 1: The pre-copy migration's performance overhead. The 60s pre-copy latency indicates a failed migration.

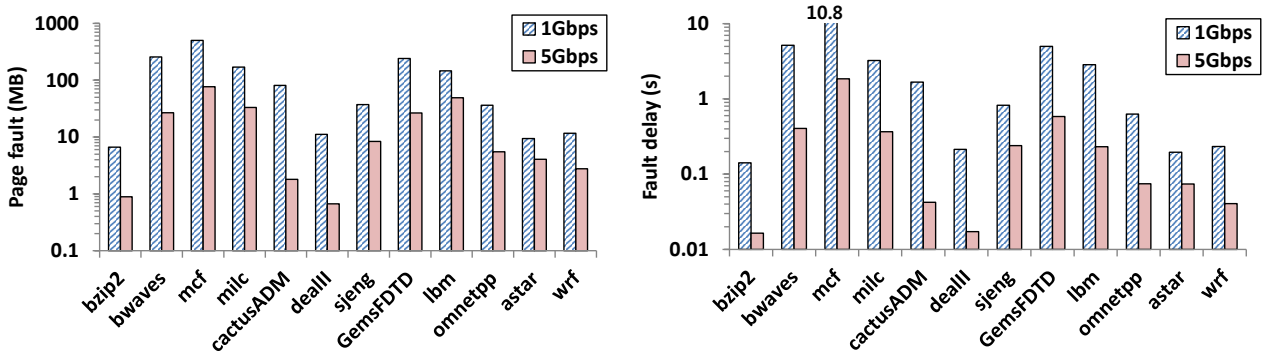
2. LIMITATION OF EXISTING SCHEMES

To motivate our guide-copy scheme, this section discusses the key limitations of conventional VM migration schemes failing to provide a fast and silent migration.

2.1 Pre-copy Migration

On a migration request, the pre-copy migration schemes [3, 7, 11, 17] copy the memory state of the VM at the source node to the destination node, while executing the VM context at the source node. When the memory copy is completed, the migration manager identifies newly updated memory pages during the copy period, and estimates the latency of transferring those pages given the network bandwidth available. If the transfer latency is below a certain threshold, the migration manager stops the VM at the source node, transfers the newly updated pages to the destination node, then transfers the VM context to the destination node, and resumes the execution. If the downtime latency is not tolerable, the migration manager must repeat the pre-copy process hoping for downtime below the threshold next time. This scheme has been widely implemented for various VM platforms such as KVM [17], Xen [3], and VMware [11].

The main problem of the pre-copy scheme occurs when the migration manager cannot find a cost-effective migration point. If the migration keeps failing to meet the given deadline (e.g., 30–60 seconds) due to too many newly updated pages, the machine is forced to stop to transfer whatever amounts of newly updated pages or the migration is simply aborted. Furthermore, the repeated pre-copy attempts consume the entire network bandwidth available. Figure 1 shows the pre-copy latency of write-intensive workloads from SPEC CPU2006 benchmarks using the KVM's default pre-copy scheme with a 60-second deadline. As the VM's memory size is 2GB, the ideal pre-copy latency (i.e., one-time memory transfer) is ~16s via 1Gbps



(a) Number of remote page faults (amount = 4KB * faults)

(b) Delay of remote fault handling

Figure 2: The post-copy migration's performance overhead as the number of page faults (4KB page) and their service latency.

network and ~3.2s via 5Gbps network. However, the figure shows that many workloads repeat lots of pre-copy attempts by taking longer than ~16s for 1Gbps network and ~3.2s for 5Gbps network. The 60s latency indicates that the migration has failed. For example, `mcf`, `milc`, and `lbm` cannot be migrated within the 60s deadline, even by consuming the entire 5Gbps network bandwidth.

2.2 Post-copy Migration

On a migration request, the post-copy migration schemes [14, 15] transfer the VM context to the destination node immediately, resume the VM execution with the memory state still present at the source node, and then initiate a background memory copy process so that the VM context at the destination node finds requested memory pages in its local memory. The background copy process is implemented as either a linear copy scheme transferring memory pages in the order of addresses, or a pivot copy scheme transferring memory pages physically closer to the recently requested pages [14, 15]. If the VM context does not find a requested memory page in its local memory, it triggers a remote page fault handler to obtain the page from the source node. Therefore, if a workload is not memory intensive or the background copy delivers requested pages in time, the post-copy scheme does not expose the machine downtime to the user. Another advantage of the post-copy scheme is the bounded number of memory page transfers, which makes it more silent than the pre-copy scheme.

The main problem of the post-copy scheme is that many remote page faults incur a significant performance loss. If a workload is memory intensive and its memory access patterns do not exhibit spatial and temporal localities, the background memory copy process becomes ineffective even though it consumes all the network bandwidth available. Figure 2 shows (a) the number of remote page faults triggered by the VM and (b) the latency to deliver the requested pages via 1Gbps and 5Gbps network. For this experiment, we carefully implement a baseline post-copy scheme with a pivot-based background copy on top of KVM. The figure shows that many SPEC CPU2006 benchmarks (e.g., `bwaves`, `mcf`, and `GemsFDTD`) suffer from a large number of remote page faults, incurring a prohibitive multi-second downtime using 1Gbps network and non-trivial downtime (e.g., 300ms–500ms) even using 5Gbps network. Note that the background copy consumes the entire network bandwidth which incurs a significant performance overhead to other network-intensive VMs in the cloud. More detailed performance overhead analysis is provided in Section 5.

3. GUIDE-COPY MIGRATION

This section describes the design goals and architecture of the guide-copy migration, and how the guide-copy scheme enables a fast and silent migration capability.

3.1 Design Goals and Requirements

We describe the key design goals and requirements of the guide-copy scheme as follows.

Guaranteed migration with a bounded memory transfer. Similar to the post-copy scheme, the guide-copy scheme transfers the VM context immediately on a migration request and then copies the remaining memory image. In this way, the guide-copy binds the amount of memory transfer as the memory size of a single VM and also guarantees a successful migration.

Isolated simultaneous execution. To obtain the future memory access pattern of the VM just migrated to the destination node, the guide-copy scheme continues to run the remaining VM context (i.e., guide context) using the memory image currently present at the source node. The guide context runs only to obtain the future memory access patterns without affecting the execution of migrated context at the destination node. Therefore, the guide context's memory and I/O accesses should not affect the execution of migrated context, and vice versa.

Adaptive in-time memory transfer. Based on the future memory access information, the guide-copy scheme copies to the destination node only the memory pages accessed by the migrated context in the near future. The simultaneous execution model arranges both VM contexts to follow the same execution flow. If the memory pages are delivered in time, the migrated context does not experience any performance loss by finding the requested pages in its local memory. As the guide context triggers on-demand memory transfers, the memory transfer rate is adaptively controlled by the application's memory access pattern.

Lightweight background memory transfer. The

guide-copy scheme also maintains a background memory copy process similar to that of the post-copy scheme, but mainly to guarantee the termination of memory transfer. This process consumes only a small portion of network bandwidth (e.g., 10% of total bandwidth), while the guide context does not experience memory accesses (e.g., non-memory intensive periods). The adaptive in-time memory copy process gets a higher priority over the background memory copy.

Remote page fault handling. When the migrated context does not find the requested memory page in its local memory due to either inaccurate or slow page deliveries, it triggers a remote page fault to bring up the page from the source node. As two VM contexts maintain their own memory images, the remote page handler should bring the original version of the memory page if it has been updated by the guide context. The number of remote page faults determines the performance overhead of migrations.

Early guide-copy termination. To avoid the guide context's unnecessary CPU usage during non-memory intensive periods, the guide-copy scheme releases the guide context, and runs as a baseline post-copy scheme once a minimum amount of memory state (e.g., 50%) is transferred and the guide copy rate is below a threshold (e.g., <1% network usage) for a long time, indicating non-intensive memory access patterns.

Guide-copy vs. other schemes. Table 1 compares the guide-copy scheme with the pre-copy and post-copy schemes. In summary, only the guide-copy scheme provides a fast and silent migration by bounding the amount of memory transfer, transferring performance-critical memory pages in time, and limiting the memory transfer rate. The guide-copy's adaptive network transfer and early CPU termination features also avoid unnecessary resource contentions.

3.2 Architecture

Figure 3 shows an example guide-copy architecture implemented on top of KVM hypervisor to achieve the design goals and requirements described in Section 3.1.

3.2.1 Components

The guide-copy architecture can be easily implemented on top of any existing VM hypervisors by adding following three components colored in gray in Figure 3(a).

Memory access log. The hypervisor running a guide context maintains a memory access log to distinguish newly accessed pages from already accessed pages during a guided execution. The memory access log is a small page table addressed by the physical address to record whether a page has been accessed by the guide context. The hypervisor uses the memory access information to trigger an in-time page delivery to the migrated context through a migration manager. The memory access log can be easily implemented as modern hypervisors already maintain a similar page table structure for the VM-to-physical memory mapping.

Migration manager. The migration manager takes three different actions. First, it copies newly accessed pages marked by the memory access log to the migrated context to avoid performance-critical remote page faults. Second, it initiates a lightweight background copy to complete the memory copy within a deadline using a small amount of network bandwidth. The migration manager prioritizes the transfer of the pages guided by the memory access log over the background copy process. Finally, the migration managers at the source and the destination nodes handle remote page faults.

Memory mapper. The destination node maintains a memory mapper to distinguish not-yet copied pages from already copied pages during a guided execution. The memory mapper has a similar page table structure to the memory access log. When the migrated context does not find a requested memory page in the memory mapper, it triggers a remote page fault through the migration manager.

Table 1: Guide-copy vs. pre- and post-copy migrations.

	Pre-copy	Post-copy	Guide-copy
Migration performance			
Migrated VM	Medium	Low	High
Migration features			
Migration guarantee	No	Yes	Yes
Bounded memory transfer	No	Yes	Yes
Immediate context transfer	No	Yes	Yes*
High-performance migration	No	No	Yes
Low-bandwidth migration	No	No	Yes
Contention (source node)			
CPU usage	High	Low	Low*
Memory bandwidth	High	Medium	Low
Network bandwidth	High	Medium	Low

*: With the early termination feature enabled

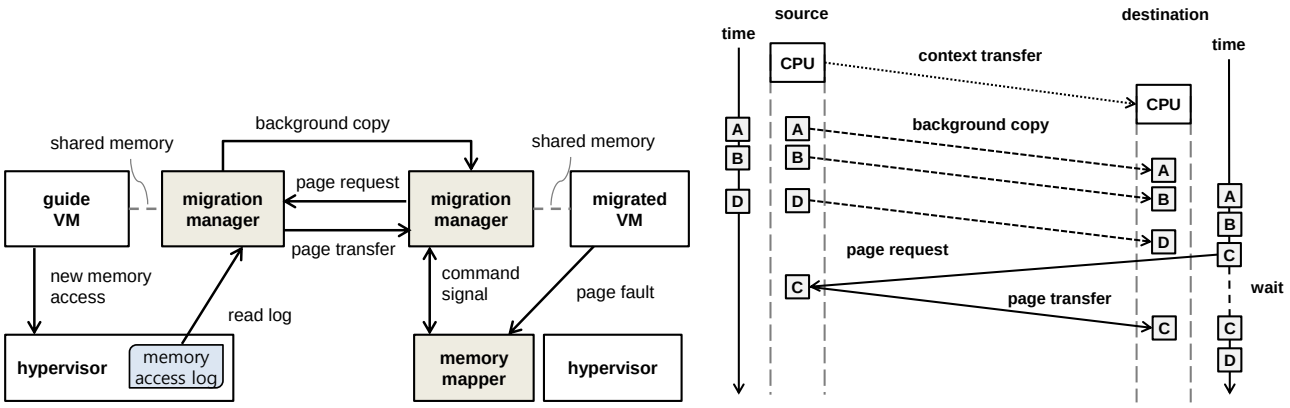
3.2.2 Guided Memory Transfer

The key intuition of the guide-copy scheme is that both the guide context and the migrated context are likely to access the same memory pages in the same order as long as the same VM context and the memory state are initially provided. Based on this intuition, the guide-copy scheme enables an accurate in-time page transfer for a fast migration and an adaptive lightweight background memory copy for a silent migration. Figure 3(b) shows an example guided memory transfer mechanism which timely delivers memory page A, B, and D in the order accessed by the guide context. In this frequent case, the migrated context at the destination node does not experience any performance loss.

When the migrated context does not find a newly requested page in its local memory (e.g., page C) due to either a failure of in-time delivery (e.g., a burst of memory access, CPU performance variation) or a deviation in the access pattern (e.g., context switches), it triggers a remote page fault. However, the late delivery is rare incurring only a small overhead to the overall performance, and the context switching is eventually compensated in a long run. Besides, the guide-copy scheme acts as a lightweight post-copy scheme during these periods. Our evaluation in Section 5 shows their impacts are minimal.

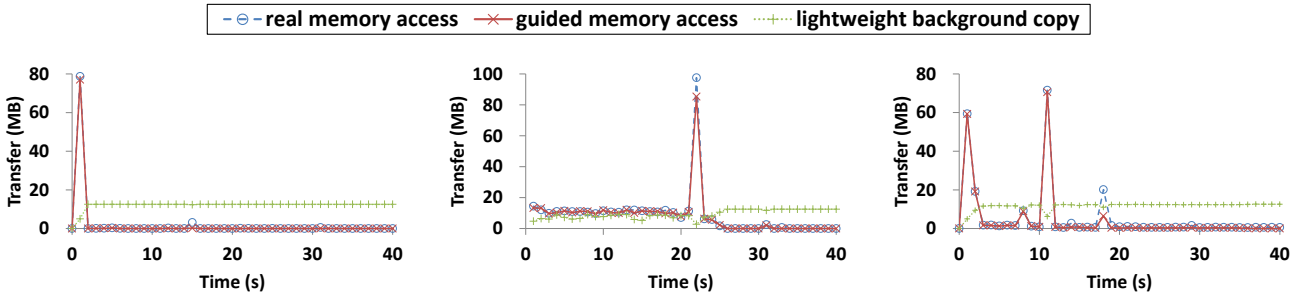
3.2.3 Isolated Simultaneous Execution

The simultaneous VM execution requires a careful handling to initially provide the same memory, disk and I/O states, and to isolate each VM's execution from the other VM's. To provide the memory isolation, the guide context wanting to write a not-yet copied page performs a copy-on-write operation, which stores its original memory content



(a) Guide-copy architecture (b) Guided memory transfer mechanism

Figure 3: The guide-copy migration's architecture with an example of a guided memory transfer scenario.



(a) Network - leslie3d (b) Network - bzip2 (c) Network - xalancbmk

Figure 4: The amount of network traffic generated by the guide-copy scheme's adaptive mode.

to another memory region, before updating the page. In this way, the migration manager handles in-time page deliveries, background copies, and remote page faults safely. The performance impact of the copy-on-write operation is minimal because (1) a write operation is not performance critical in general, (2) we run the copy-on-write process as a background process, and (3) the page-level copy latency can be hidden further by copying the to-be-updated block first.

The guide-copy scheme treats disk updates as memory updates using the VM's existing disk-cache memory. We modify the existing disk cache mechanism to be flexible so that a disk cache overflow or eviction can be avoided. In this way, we provide an isolated view of the disk update to each VM. We implement our example guide-copy scheme on top of an Network File System (NFS), but our scheme can be implemented on any file system as long as both VMs can access to the initial state of disk drive. Disk updates can avoid copy-on-write operations if the disk cache is flushed on a migration request. Finally, only the migrated VM at the destination node sends its network messages to the outside, while the incoming messages are delivered to both VM contexts.

3.2.4 *Adaptive Network Consumption*

The guide-copy's migration manager transfers only the memory pages likely to be used at the destination node in the near future based on the memory access log. This access-pattern based memory transfer mechanism reduces the peak network usage rates as the network bandwidth is adaptively controlled by the guide context. To compensate the guide-controlled transfer, the manager also initiates a pivot-based background memory copy using only a small portion of network bandwidth (e.g., 10% of total network bandwidth). The pivot copy prioritizes transferring memory pages physically closer to the recently requested pages [14, 15]. The manager gives a higher priority to the adaptive guide memory transfer over the pivot-based background memory transfer. The background memory transfer completes the migration of non-memory intensive workloads.

Figure 4 shows how deploying the guided copy and the lightweight background copy together effectively removes most of the performance-critical remote page faults and also minimizes the peak network usage rates. The figure shows the amount of network traffic generated by three SPEC CPU2006 workloads during a guide-copy migration period. They show that the guided transfer correctly predicts the future memory accesses, while completing the migration with the background copy consuming only 10% of 1Gbps network bandwidth.

3.2.5 *Early Guide-copy Termination*

For non-memory intensive workloads, the guide-copy scheme might have to run the guide context without transferring many memory pages. To avoid the guide context's unnecessary CPU usage during non-memory intensive periods, the

guide-copy scheme implements a feature to release the guide context when the guided execution is no longer cost-effective. Figure 4 already hints that the in-time memory transfer is the most effective during the initial phase of the migration and the background memory copy will cover later peaks (e.g., `xalancbmk`). Therefore, the guide-copy scheme releases the guide context, and runs as a baseline post-copy scheme once the amount of memory transferred so far is larger than a given threshold (e.g., 50%) and the guide copy rate is below a threshold (e.g., <1% network usage) for a long time, indicating non-intensive memory access patterns. Section 5 evaluates the actual impact of early guide-copy termination.

4. EXAMPLE GUIDE-COPY IMPLEMENTATION

In this section, we implement our example guide-copy scheme in KVM [17] (kvm-kmod 2.6.39 and qemu-kvm 0.14.1). We describe our baseline post-copy and guide-copy implementations as designed in Section 3.

4.1 Baseline Post-copy Scheme

We implement our post-copy scheme based on [15]. We first implement the migration manager and the memory mapper in KVM, shown in Figure 3(a). When the VM at the destination node does not find a newly requested page in its local memory, the migration manager receives the memory page from the source node by triggering a remote page fault handler. At the same time, the migration manager at the source node initiates a pivot-based background copy process. We keep three to seven pivots to transfer memory pages physically closer to the three to seven most recently requested regions. The number of pivots is carefully chosen to obtain the best performance of existing post-copy scheme for a fair comparison in Section 5. The baseline post-copy scheme performs the following procedure:

1. **VM memory mapping.** During the post-copy period, the migrated VM accesses its memory mapper by exploiting `mmap()` function to detect a page fault.
2. **Page fault detection.** When the migrated VM accesses a new memory page not found in its memory mapper, the memory mapper detects a remote page fault.
3. **Page fault request.** The migration manager at the destination node sends the remote page fault request to the migration manager at the source node.
4. **Page reception.** The migration manager at the destination node receives the corresponding page from the migration manager at the source node.
5. **Page fault resolution.** The migration manager at the destination node updates the memory mapper to mark that the page has been correctly received.

Our post-copy implementation might not achieve the best performance as it uses the extra processes (e.g., the migration manager) instead of incorporating them directly into the KVM module. However, this implementation is sufficient to show our scheme's potential.

In addition, when the VM context is initially moved from the source node to the destination node, the migration manager must move some critical states such as pinned pages (e.g., `virtio` [28] and network (`RTL8139`) ring buffer [27] descriptors).

4.2 Guide-copy Capability

Based on the baseline post-copy mechanism, we implement our guide-copy capability by adding the memory access log and extra features to the migration managers to enable the isolated simultaneous execution, the guided memory page transfer, the lightweight background copy, and early termination. The modern systems support a hardware virtualization with a nested paging capability to avoid unnecessary hypervisor interferences [5]. We exploit the existing structure to implement our guide-copy mechanism easily. The detailed actions are taken as follows:

1. **Flush the nested page table.** On a migration request, we flush the VM's nested page table to detect newly accessed memory pages.
2. **New memory access detection.** The hypervisor newly constructs a nested page table with respect to the locations of newly accessed memory pages. For a newly accessed page, the hypervisor performs a copy-on-write operation to store the original content at a different memory region and then updates the page. It also records the address in the memory access log.
3. **Guided memory transfer.** The migration manager transfers the original memory pages whose addresses are recorded in the memory access log.

The performance overhead of flushing the nested page table is negligible because it occurs only once per migration. The overhead of remote page faults also outweighs the flush overhead. In addition, we parallelize KVM's page fault handler to support an increasing number of remote page faults generated by multi-thread workloads.

5. EVALUATION

We evaluate our guide-copy migration scheme on our in-house datacenter consisting of Intel Xeon X5650 nodes connected by the network bandwidth of 1Gbps and 5Gbps. We modify QEMU-KVM 0.14.1 to support the guide-copy migration scheme. Each node runs the modified QEMU-KVM 0.14.1 on top of Ubuntu 10.04 64b host operating system. The QEMU-KVM constructs a VM image with two virtual CPUs and 2GB RAM on Ubuntu 10.04 64b guest operating system. Due to the non-deterministic experimental environment, all results are provided with the 95% confidence error bars.

5.1 Workload Characteristics

To evaluate our example guide-copy implementation, we select a subset of SPEC CPU2006 [13], PARSEC [6], and CloudSuite [9] benchmarks which provides varying memory access behaviors, multi-thread behaviors, and real-world datacenter behaviors, respectively. Table 2 sorts the memory-intensive workloads based on the maximum new memory access rates since a migration starts after a 10-second warm up period. The peak new memory access rate determines the peak network bandwidth required by the post-copy and guide-copy schemes, while the peak memory write rate indicates the pre-copy scheme's repeated memory transfer and unbounded amount of network usage. We do not evaluate non-memory intensive workloads in this paper as they do not affect the performance of migration schemes. In addition, we do not evaluate network message-intensive workloads because the input message synchronization is beyond the scope of this paper. This limitation is orthogonal to the guide-copy scheme's fast and

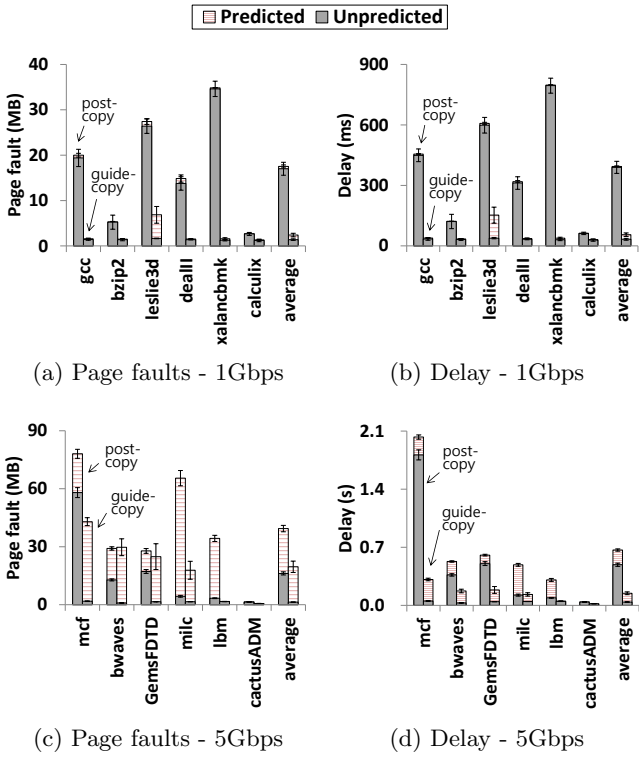
silent migration capability.

5.2 Performance: Reduced Faults and Delay

As the performance of VM migrations always improves with a faster network, we classify SPEC CPU 2006 workloads into two groups based on their memory access intensity: a group using 1Gbps network (i.e., `gcc`, `bzip2`, `leslie3d`, `dealII`, `xalancbmk`, and `calculix`) and the other using 5Gbps network (i.e., `mcf`, `bwaves`, `GemsFDTD`, `milc`, `lbm`, and `cactusADM`).

Figure 5 shows the number of remote page faults generated by the post-copy and their fault handling latencies. For this experiment, we classify remote page faults into two types: predicted faults and unpredicted faults. The predicted faults indicate the correctly predicted faults of the post-copy's pivot-based prediction and the guide-copy's future memory access prediction. Therefore, transferring the predicted faults in time removes remote page faults at the destination node, while the unpredicted faults trigger unavoidable remote page fault deliveries. Figure 5(a) and (b) show that the guide-copy scheme correctly predicts most of the page faults (i.e., up to ~85%) and reduces the performance overhead correspondingly for the workloads using 1Gbps network.

As shown in Figure 5(c) and (d), the experiments using 5Gbps network show a similar level of prediction accuracy. As these workloads are highly memory intensive, even a correctly predicted fault can incur a delay due to TCP buffering. However, the performance impact of



(a) Page faults - 1Gbps (b) Delay - 1Gbps

(c) Page faults - 5Gbps (d) Delay - 5Gbps

Figure 5: Guide-copy's in-time memory transfer reducing the number of page faults and their service latency.

Table 2: Selected memory-intensive workloads sorted by the peak of new memory access rates.

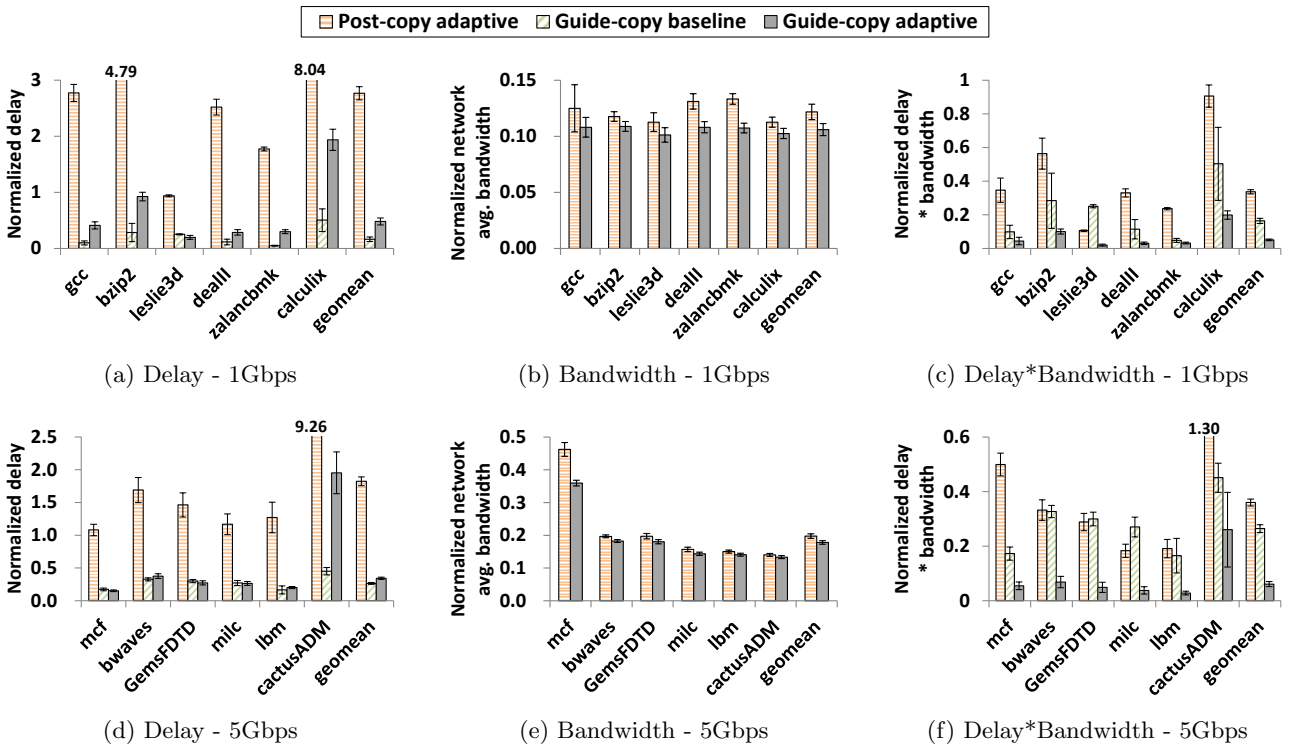
	working set (MB)	peak write rate (MB/s)	peak new access rate (MB/s)
SPEC CPU2006			
mcf	1673.1	1120.5	975.3
bwaves	875.9	262.7	776.3
GemsFDTD	811.6	759.1	716.6
milc	668.8	518.2	430.4
lbm	429.7	404.7	424.8
cactusADM	418.5	108.5	293.4
gcc	243.0	165.0	99.8
bzip2	353.1	285.8	98.7
leslie3d	83.6	65.3	79.4
dealII	188.0	169.1	65.1
xalancbmk	204.7	126.7	59.4
calculix	123.8	87.0	57.7
PARSEC			
canneal	943.6	605.5	938.8
freqmine	728.4	58.1	381.5
blackscholes	285.5	7.0	268.9
x264	1337.8	177.3	184.9
fluidanimate	178.4	93.5	175.8
facesim	180.2	104.5	171.6
dedup	918.3	173.5	136.2
streamcluster	110.4	11.0	106.8
CloudSuite			
Data Analytics	2021.8	826.9	763.2
Data Serving	297.3	145.7	148.8
Software Testing	88.9	77.9	48.9

TCP buffering does not affect the guide-copy's performance improvement, as shown in Figure 5(d), because those remote page faults are serviced much faster. Therefore, the guide-copy scheme reduces the performance overhead by ~80% on average for the workloads using 5Gbps as well.

To see overall migration delays during the execution of workloads, we perform back-to-back migrations [24] as shown in Figure 6. For three memory intensive workloads, we keep migrating the VM with a 5s interval until the workload completes. For this experiment, `mcf`, `bwaves`, and `GemsFDTD` migrate 75 times, 90 times, and 97 times, respectively. The results show that our guide-copy implementation removes the performance overhead of the baseline post-copy migration significantly (i.e., 50s–100s).

5.3 Silence: Reduced Network Usage

Figure 7 shows that the migration delay increases when we decrease the network bandwidth from 1Gbps or from 5Gbps. In this experiment, the guide-copy scheme prioritizes the guided copy over the background copy, but uses the remaining network bandwidth for the background copy process. The results indicate that the guide-copy scheme outperforms the post-copy scheme significantly even with much smaller network bandwidth. For example, both `bzip2` and `cactusADM` show that the guide-copy scheme requires only 40% of static bandwidth assignment to achieve the 81%–85% of page fault reduction, as shown in Figure 5. Therefore, this result shows the guide-copy scheme's high potential of the silent migration and motivates its adaptive execution model which dynamically assigns the network



(a) Delay - 1Gbps (b) Bandwidth - 1Gbps (c) Delay*Bandwidth - 1Gbps

(d) Delay - 5Gbps (e) Bandwidth - 5Gbps (f) Delay*Bandwidth - 5Gbps

Figure 8: Guide-copy's cost-effective adaptive migration using 1Gbps and 5Gbps network bandwidth. (normalized to the baseline post-copy scheme)

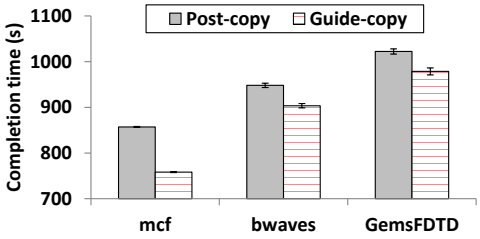
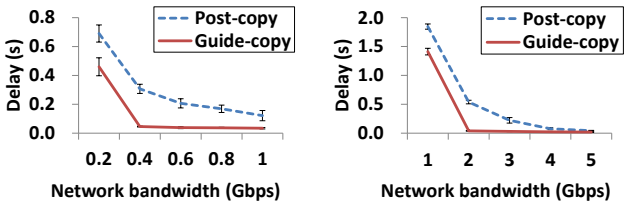


Figure 6: The execution time of workloads repeating back-to-back post-copy and guide-copy migrations with a 5s interval.



(a) Delay - bzip2 (b) Delay - cactusADM

Figure 7: The guide-copy migration delay with varying network bandwidth availability.

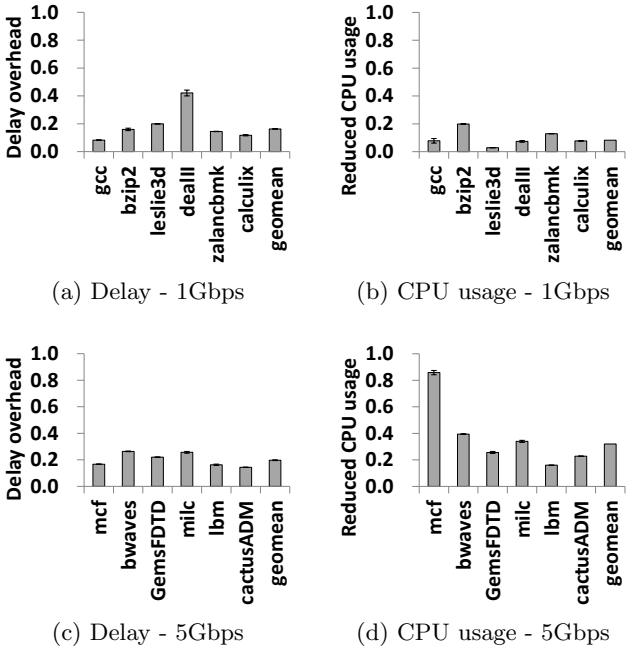
bandwidth while limiting the bandwidth given to the background copy process.

5.4 More Silence: Adaptive Network Usage

As shown in Figure 4, the guide-copy scheme can control the network bandwidth usage by prioritizing the in-time guided transfer over the lightweight pivot-based background copy. For this experiment, we implement *guide-copy adaptive* which provides the requested amount of bandwidth to the guided transfer, but limits the network bandwidth used by the background copy to 10% of network bandwidth. For a fair comparison, we also implement *post-copy adaptive* which controls the speed of transferring pages near each pivot point. For example, the post-copy adaptive transfers 50–200 memory pages around a new pivot for 1–5Gbps network. Therefore, the post-copy adaptive scheme experiencing many page faults performs like a full-bandwidth post copy scheme, whereas it performs like the guide-copy adaptive scheme's lightweight background copy for infrequent page faults using 10% of network bandwidth.

Therefore, both adaptive schemes provide more silent migrations than their baseline schemes at the cost of decreasing page fault prediction accuracy. Figure 8 compares the performance and the average network usage of post-copy adaptive, guide-copy baseline, and guide-copy adaptive schemes. We divide the total network usage by the whole migration period to obtain the average network usage, which indicates the degree of network disturbance during the period. All results are normalized to the baseline post-copy scheme. Figure 8(a) and (d) show the increasing migration latency, whereas Figure 8(b) and (e) show the reducing average network usage as expected.

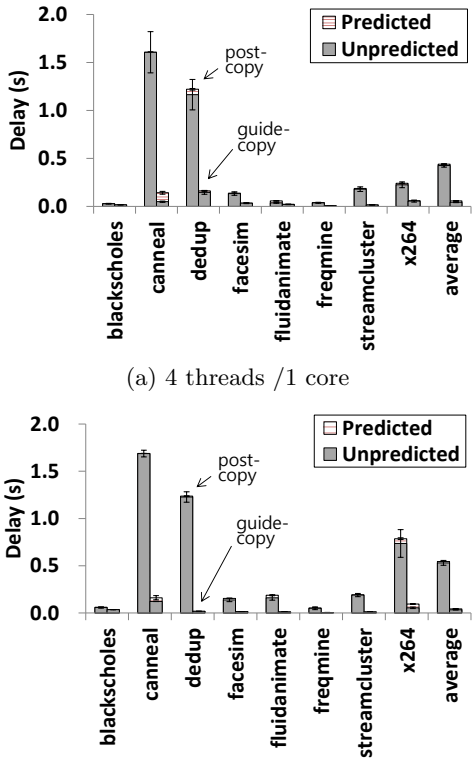
To compare the overall cost-effectiveness, we compute the product of normalized delay and average bandwidth usage



(a) Delay - 1Gbps (b) CPU usage - 1Gbps

(c) Delay - 5Gbps (d) CPU usage - 5Gbps

Figure 9: The impact of early guide-copy termination. (normalized to the post-copy's adaptive mode)



(a) 4 threads /1 core

(b) 4 threads /4 cores

Figure 10: Guide-copy's performance against multi-threaded workloads using 4 threads.

as shown in Figure 8(c) and (f). These results indicate that the guide-copy adaptive scheme outperforms both the post-copy adaptive scheme and the guide-copy baseline scheme significantly as it reduces the most of average bandwidth usage with only a negligible performance overhead. The results show that the guide-copy adaptive is the only scheme to enable a fast and silent migration for highly utilized datacenters.

5.5 Early Guide-copy Termination

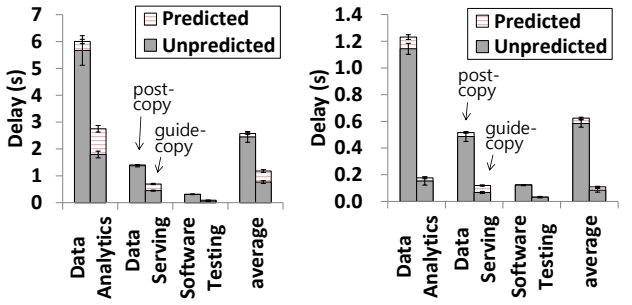
As discussed in Section 3.2.5, the guide-copy scheme might have to run the guide context without transferring many memory pages. To avoid such unnecessary CPU usage during non-memory intensive periods, the guide-copy scheme releases the guide context instead of wasting the CPU until the background copy completes the migration. For this experiment, we terminate the guide-copy execution when the guide context accesses less than 3MB of memory for the last three seconds.

Figure 9 illustrates the impact of early CPU termination as the increased performance overhead and the reduced CPU usage. The increased overhead and the reduced CPU usage are normalized to those of post-copy's adaptive mode. The results show that the early guide-copy execution reduces the wasted CPU usage, while still outperforming the post-copy's adaptive mode. The impact of early termination is greater for a slow network causing a longer migration period. Heavily memory intensive workloads such as `mcf` do not allow an early termination as they keep transferring memory pages. As different thresholds will lead to different tradeoffs in terms of the performance overhead and the reduced CPU usage, we believe it is worth trying adaptive thresholds, which we leave as our future work.

5.6 Multi-threaded Workloads

For this experiment, we evaluate the potential of the guide-copy scheme against PARSEC, a widely used suite of multi-threaded workloads, for which each workload exercises various memory sharing and synchronization behaviors by running multiple threads. As discussed in Section 3.2.2, the guide-copy scheme might suffer from frequent context switches. Figure 10 compares the migration performance of the post-copy scheme and the guide-copy scheme. For Figure 10(a), we configure each workload to deploy four threads on a single physical core to maximize the impact of context switches. For Figure 10(b), each workload deploys four threads on four physical cores to reduce the impact of context switches. To focus on the impact of context switches, we use the 5Gbps network to reduce the impact of network speed.

Figure 10(a) shows that the guide-copy scheme outperforms the post-copy scheme by 87% on average by transferring the most of pages in time. We observe that the context switches can be compensated in a long run as the transferred pages are eventually accessed by the corresponding thread or other threads sharing the pages. In addition, the lightweight background copy effectively reduces many remote page faults. Furthermore, if we assign a single thread per core to reduce the impact of context switches, the guide-copy scheme outperforms the post-copy scheme by 92%, as shown in Figure 10(b). This result indicates a high potential of arranging the guide context to perform similar context switches as the migrated context, even though it is not



(a) Delay - 1Gbps (b) Delay - 5Gbps

Figure 11: Guide-copy's performance against CloudSuite.

required. If necessary, synchronizing two VM contexts can be implemented if the migrated context informs the guide context of its context switch events. We are currently exploring a cost-effective method of implementing the context synchronization.

5.7 CloudSuite Workloads

We also evaluate the potential of the guide-copy scheme against CloudSuite, a recently introduced benchmark set consisting of open-source datacenter-friendly applications, for which each application tends to use massive data in a scalable system. Figure 11 compares the migration performance of the post-copy scheme and the guide-copy scheme. For Figure 11(a) and (b), we configure each workload on the 1Gbps network and on the 5Gbps network respectively.

Figure 11(a) shows that the guide-copy scheme outperforms the post-copy scheme by 54% on average, as the guide-copy scheme predicts memory access pattern with a high degree of accuracy. If we use the 5Gbps network to reduce the impact of network bandwidth limitation, the guide-copy scheme outperforms the post-copy scheme by 82%, as shown in Figure 11(b). The high memory access rates of CloudSuite leads to the guide-copy's increasing performance advantage using a faster network. As a summary, the guide-copy scheme presented in this paper shows the great potential to improve the migration performance of the existing scheme for a various spectrum of workloads, while consuming the minimum level of peak network bandwidth.

6. RELATED WORK

Cloud infrastructure providers reduce their maintenance costs by having many VMs oversubscribe various computing resources. However, a resource contention caused by sudden huge resource demands may incur a performance degradation [2]. In order to prevent the contention, various dynamic resource management (DRM) schemes [4, 11, 18, 22, 23, 26, 30, 33, 35] address resource contentions at CPU, memory, network, and data centers have been introduced. One key technique of the DRM schemes is a live VM migration. The live VM migration moves a VM between nodes without any noticeable performance disturbance. There are two basic live migration schemes: pre-copy and post-copy schemes.

The pre-copy scheme [3, 7, 11, 17] is the most popular live VM migration scheme, widely implemented on various VM platforms such as KVM [17], Xen [3], and VMware [11]. The pre-copy scheme transfers a VM context after copying all memory states from the source node to the destination node. The memory pages updated during the memory transfer should be retransmitted. Unfortunately, the amount of memory transfer is not bounded in write-intensive workloads, hurting the performance of other VMs in the cloud [25]. There exist several techniques to reduce the amount of memory transfer. Delta compression [32] compresses memory pages by exploiting the differences from the previously transmitted pages. The trace and replay methods [21] log operations of the source VM, and replay the logs in the destination VM. However, these techniques still generate an unbounded amount of network traffic and do not guarantee successful migrations.

The post-copy scheme [14, 15] transfers the VM context after transferring the memory states. The post-copy scheme can cause a significant performance overhead at the destination node due to slow on-demand remote page faults. In order to reduce the number of the remote page faults, the post-copy scheme initiates the background memory copy process so that the VM context at the destination node finds its memory pages in time. The background memory copy process is often implemented as a pivot copy scheme [14], which copies the memory pages closer to the requested page. However, the post-copy scheme does not work for workloads with low spatial-locality, causing significant amount of network traffic.

Existing migration schemes use the entire network bandwidth to reduce the migration delay. Dynamic self-ballooning [12, 34] reduces the amount of memory transfer by transferring only a VM context's working set. However, ballooning still uses the entire network bandwidth. Remus [8] maintains a replicated VM to perform a live migration with minimum overheads. However, the VM replication doubles the resource requirements and also suffers from synchronization overheads.

VM migration schemes often assume an NFS to eliminate the need of disk migrations. However, when there is the need (e.g., inter-data center migrations), reducing the size of disk transfer is the most important design goal. Capsule disk [29] uses a hierarchical redundant disk management scheme with a copy-on-write (COW) capability. Keeping disks redundant in advance [1, 36] reduces the size of disk transfer.

Identifying a resource contention to trigger a VM migration has been widely explored. Wood et al. [37] determine a possible contention based on the usage profiling. Other studies determine resource contention or consolidation points by monitoring resource utilization [10, 16]. On-demand elastic scaling and VM provisioning [20, 31] determine when to perform migrations. To resolve the resource contention, several approaches have been proposed. Overdriver [35] provides a network memory to survive an instant memory exhaustion. SnowFlock [19] supports a fast VM cloning which copies the memory and disk on demand.

All related studies above do not provide a fast and silent migration. However, they can be applied on top of our guide-copy scheme for added benefits.

7. CONCLUSION

Cloud infrastructure providers require a fast and silent migration scheme to maintain the SLAs between all users

and providers, while operating data centers at the minimum costs. Given the increasing demand on highly-utilized cloud, a fast and silent migration will be of paramount importance. However, existing VM migration schemes are not suitable for such highly-utilized clouds by incurring a significant performance overhead to both the migrating VM and other VMs running in the cloud, due to the latency- and bandwidth-critical memory transfer mechanism. The guide-copy migration proposed in this paper provides a fast and silent migration by transferring only the minimum amount of memory pages in time. Our guide-copy migration's highly accurate and low-bandwidth memory transfer provides a fast and silent migration capability, making it a promising VM migration technique for future clouds.

8. ACKNOWLEDGMENTS

This research was supported by the NRF Grant funded by the Korean Government (MSIP) (2011-0014817) and by the MSIP under the "IT Consilience Creative Program" (NIPA-2013-H0203-13-1001) and the ITRC (NIPA-2013-H0301-13-3002) support program supervised by the NIPA.

9. REFERENCES

[1] A. Anand, V. Sekar, and A. Akella. SmartRE: an architecture for coordinated network-wide redundancy elimination. In *SIGCOMM'09*.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4), Jan. 2010.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP'03*.

[4] M. N. Bennani and D. A. Menascé. Resource allocation for autonomic data centers using analytic performance models. In *ICAC'05*.

[5] R. Bhargava, B. Serebrin, F. Spadini, and S. Manne. Accelerating two-dimensional page walks for virtualized systems. In *ASPLOS'08*.

[6] C. Bienia, S. Kumar, J. Singh, and K. Li. The PARSEC benchmark suite: characterization and architectural implications. In *PACT'08*.

[7] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *NSDI'05*.

[8] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: high availability via asynchronous virtual machine replication. In *NSDI'08*.

[9] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *ASPLOS'12*.

[10] D. Gmach, J. Rolia, L. Cherkasova, G. Belrose, T. Turicchi, and A. Kemper. An integrated approach to resource pool management: policies, efficiency and quality metrics. In *DSN'08*.

[11] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu. VMware distributed resource management: design, implementation, and lessons learned. *VMware Technical Journal*, 2012.

[12] S. M. Hand. Self-paging in the Nemesis operating system. In *OSDI'99*.

[13] J. L. Henning. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4), Sept. 2006.

[14] M. R. Hines and K. Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *VEE'09*.

[15] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi. Enabling instantaneous relocation of virtual machines with a lightweight VMM extension. In *CCGRID'10*.

[16] G. Khanna, K. Beaty, G. Kar, and A. Kochut. Application performance management in virtualized server environments. In *NOMS'06*.

[17] A. Kivity, Y. Kama, D. Laor, U. Lublin, and A. Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, 2007.

[18] D. Kusic and N. Kandasamy. Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems. In *ICAC'06*.

[19] H. A. Lagar-cavilla, J. A. Whitney, A. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan. SnowFlock: rapid virtual machine cloning for cloud computing. In *EurySys'09*.

[20] C. Liu, B. T. Loo, and Y. Mao. Declarative automated cloud resource orchestration. In *SOCC'11*.

[21] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu. Live migration of virtual machine based on full system trace and replay. In *HPDC'09*.

[22] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis. Efficient resource provisioning in compute clouds via VM multiplexing. In *ICAC'10*.

[23] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM'10*.

[24] M. Nelson, B.-H. Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *ATC'05*.

[25] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant. Automated control of multiple virtualized resources. In *EuroSys'09*.

[26] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem. Adaptive control of virtualized resources in utility computing environments. In *EuroSys'07*.

[27] REALTEK. RTL8139(A/B) programming guide: (v0.1). 1999.

[28] R. Russell. virtio: Towards a de-facto standard for virtual I/O devices. *ACM SIGOPS Operating Systems Review*, 42(5), July 2008.

[29] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum. Optimizing the migration of virtual computers. In *OSDI'02*.

[30] J. Schad, J. Dittrich, and J. Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. In *VLDB'10*.

[31] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. CloudScale: elastic resource scaling for multi-tenant

cloud systems. In *SOCC'11*.

[32] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmroth. Evaluation of delta compression techniques for efficient live migration of large virtual machines. In *VEE'11*.

[33] B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource overbooking and application profiling in a shared internet hosting platform. *ACM Transactions on Internet Technology*, 9(1), Feb. 2009.

[34] C. A. Waldspurger. Memory resource management in VMware ESX server. In *OSDI'02*.

[35] D. Williams, H. Weatherspoon, H. Jamjoom, and Y.-H. Liu. Overdriver: handling memory overload in an oversubscribed cloud. In *VEE'11*.

[36] T. Wood, P. Shenoy, K. Ramakrishnan, and J. V. der Merwe. CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines. In *VEE'11*.

[37] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Black-box and gray-box strategies for virtual machine migration. In *NSDI'07*.